COFI: uma metodologia para geração de teste para sistemas espaciais – teoria e prática

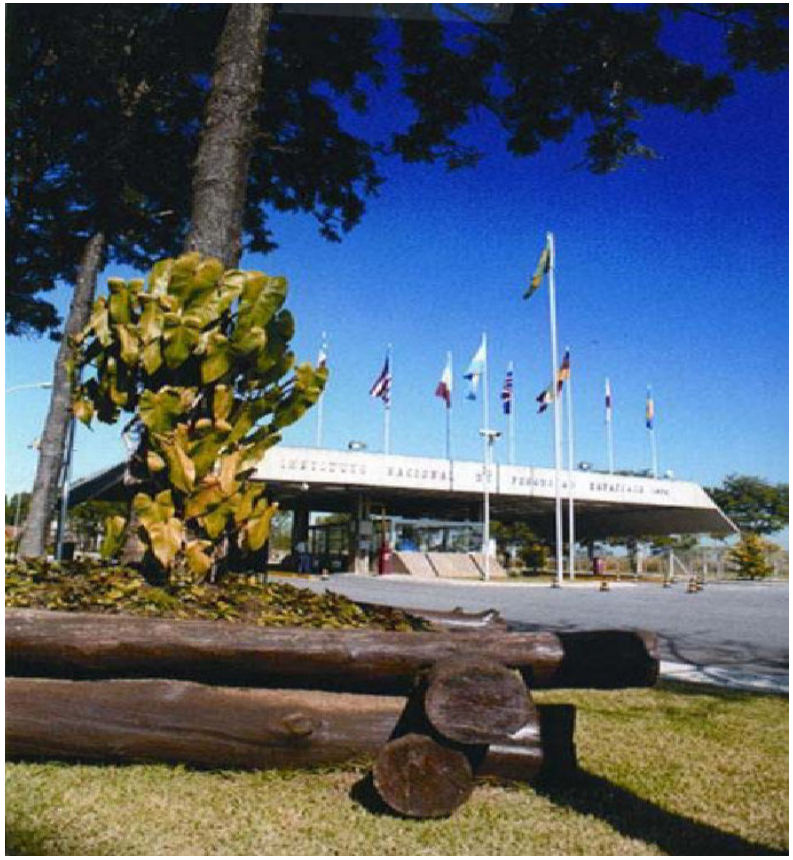
Ana Maria Ambrosio

Eliane Martins

Tópicos

- Visão geral do INPE
- Metodologia de teste CoFI
- Resultados obtidos com a avaliação da metodologia em:
  - Serviço de computador de bordo
  - Comparação com método de teste N+
  - Processo de V&V Independente para aceitação de software espacial (projeto QSEE)

Instituto Nacional de Pesquisas Espaciais

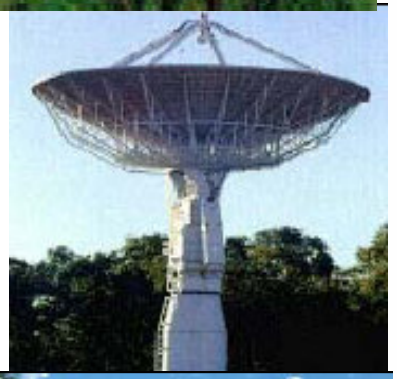


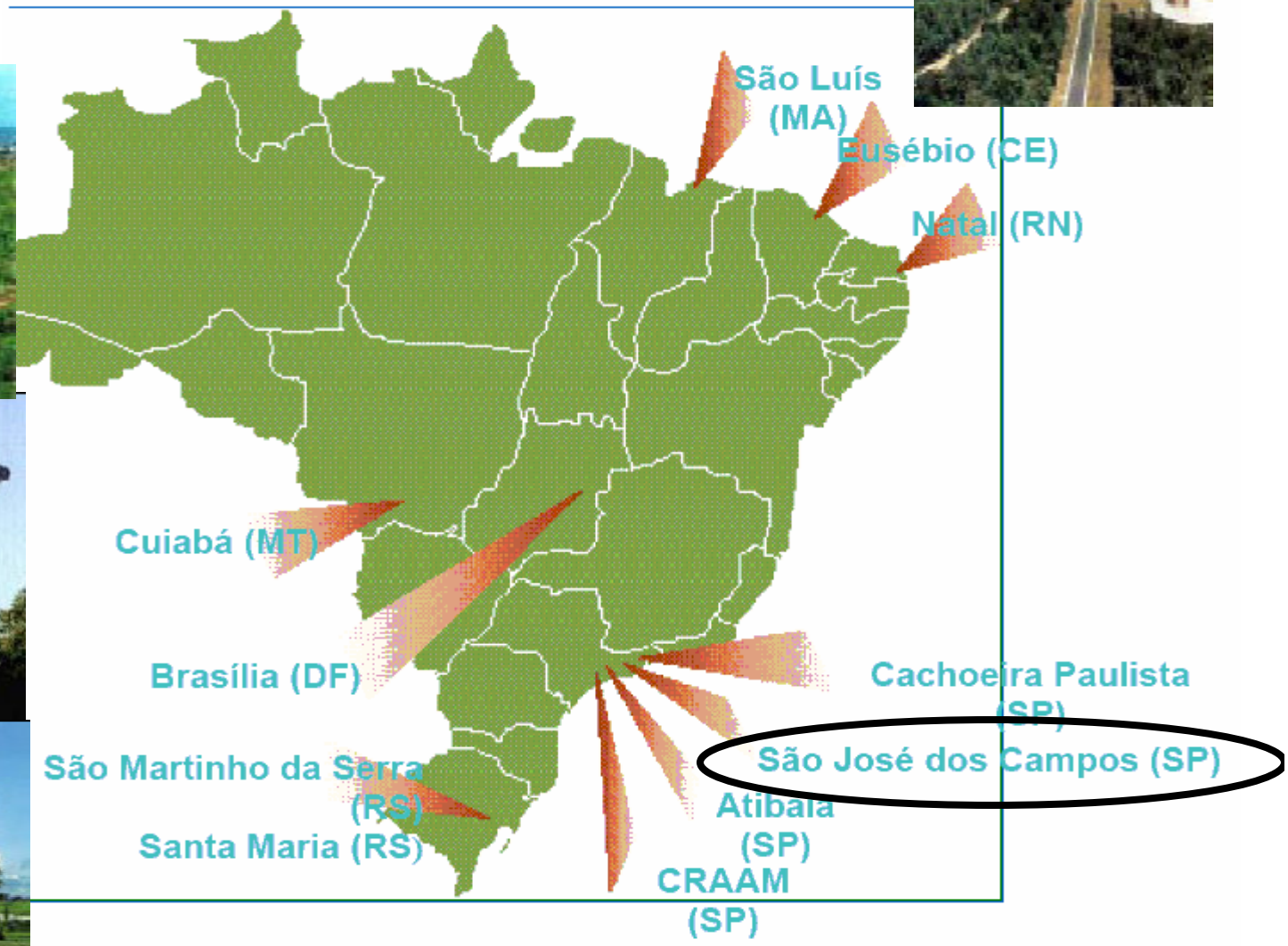
Subordinado ao Ministério de Ciência e Tecnologia - MCT

Missão: Produzir ciência e tecnologia nas áreas espacial e do ambiente terrestre e oferecer produtos e serviços em benefício do Brasil.

Fundado em 1961.

Instalações do INPE

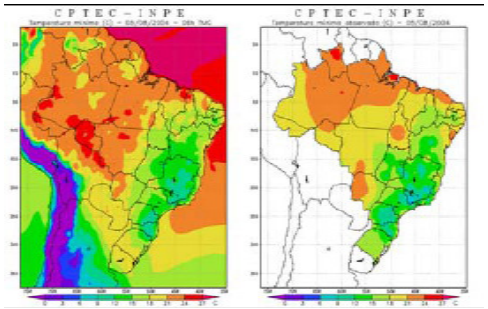




Principais áreas de atuação

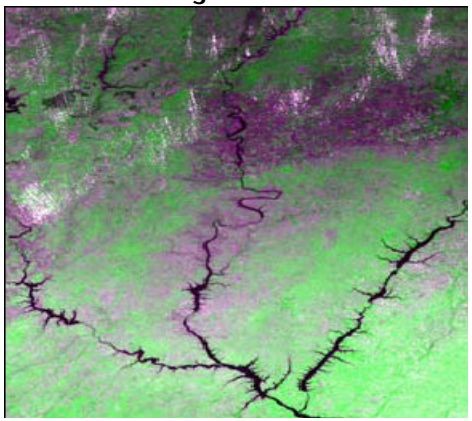
Ciências espaciais
e atmosféricas

Previsão de tempo
e estudos climáticos

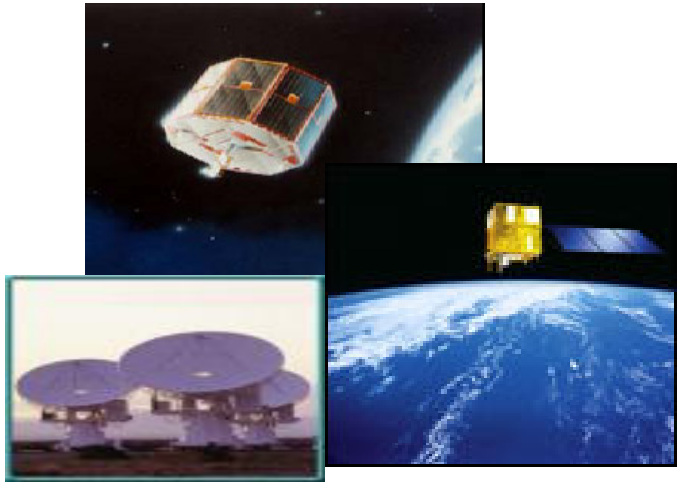


Laboratório de
Integração e Testes

Observação da Terra



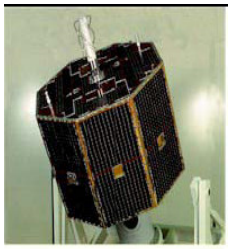
Engenharia e Tecnologia Espacial



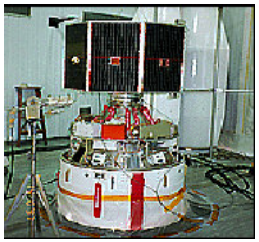
Centro de rastreio
e controle de satélites

Missões espaciais realizadas

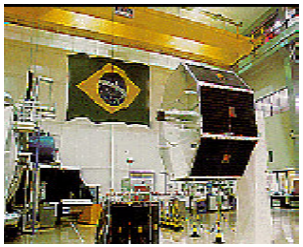
MECB – Missão Espacial Completa Brasileira



SCD1/93



SCD2/97



SCD2A/98

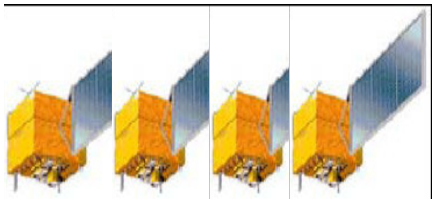
SCDav

CBERS – China-Brazil Earth Resources Satellite

CBERS1/99

CBERS2/03

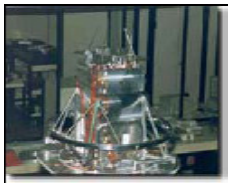
CBERS2B/07



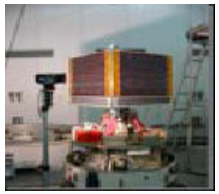
CBERS 3, 4 5 e 6

Satélites científicos e tecnológicos

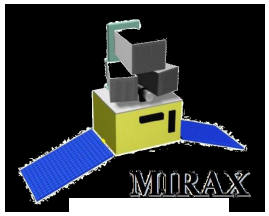
SACI-1/99



SACI-2/99



Satec/03



Mirax

FBM/02, EQUARS, ItaSat, Lattes

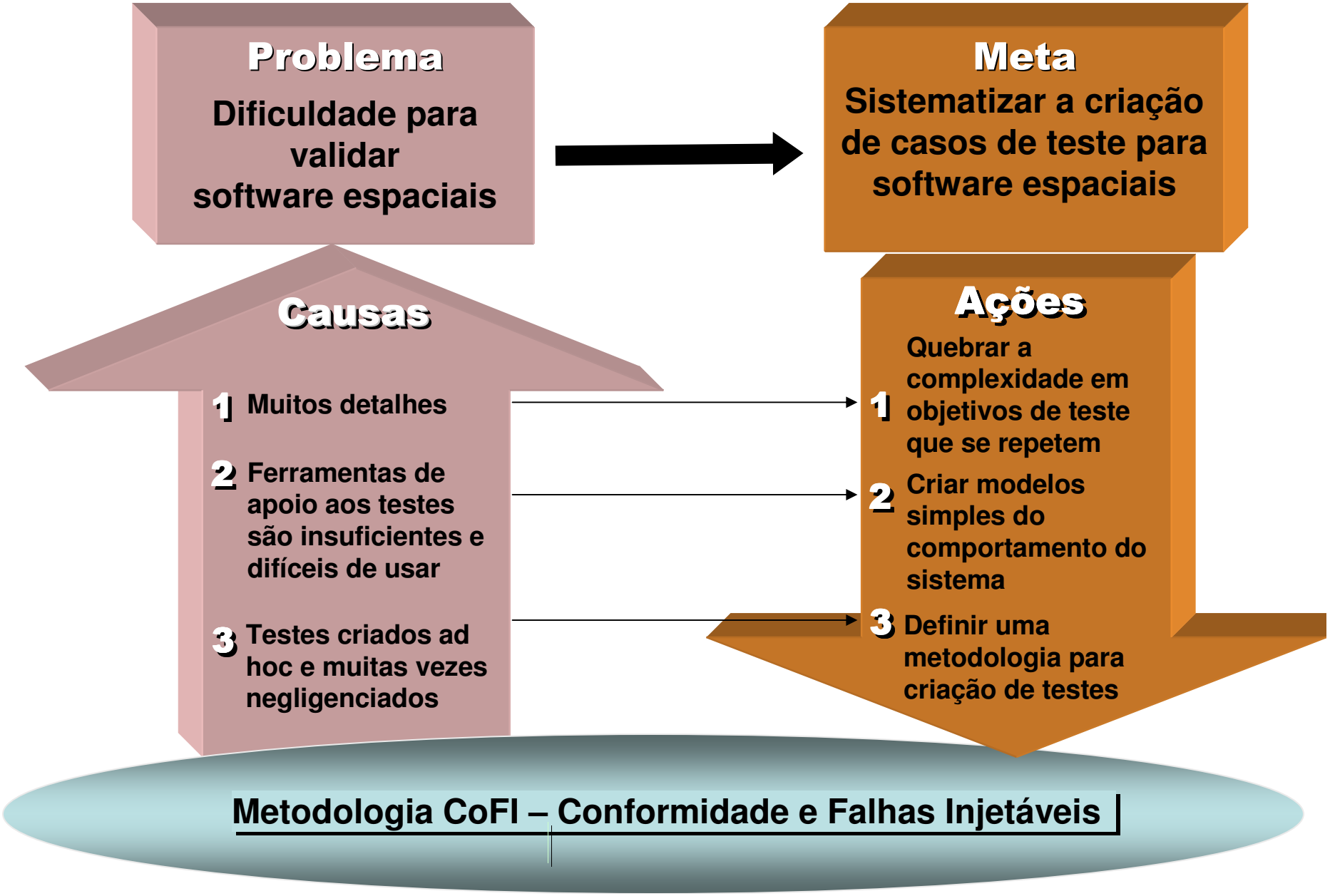
Futuros – Amazônia, MapSar, GPM, etc...

Motivação para reduzir insucessos em missões espaciais

PROJETO	INCIDENTE	CAUSA
ARIANE 501 junho 96	Falha no lançamento, perda de 4 satélites	**Insuficiência de teste do software**
MARS CLIMATE ORBITER setembro 98	Perda do veículo	Troca de unidades de medida. **Insuficiência de teste de software**
MARS P. LANDER dezembro 98	Perda do *lander*	Erro de software forçou desligamento prematuro da máquina de pouso: **insuficiência de teste de software**
TITAN-4B abril 99	Falha ao colocar o satélite militar em sua órbita final	Erro no ponto decimal no sistema de guiagem. **Insuficiência de teste apropriado no software**
DELTA-3 abril 99	Lançamento abortado	Falha na inicialização da máquina principal. **Insuficiência de teste do software de bordo**
PAS-9 março 2000	Perda do satélite ICO da *Global Communication*	Erro na atualização do software de solo. **Insuficiência de teste de software**

Fonte: Mazza, 2000

Problema x Meta



Objetivo da CoFI

Sistematizar a criação de casos de testes levando em consideração:

o uso de ferramentas de **geração automática de teste a partir de FSM/EFSM** existentes, mas evitando a explosão do número de casos de teste e a complexidade dos modelos

a necessidades de teste de software em aplicações espaciais → associando **teste de conformidade** à técnica de **injeção de falhas**

Por quê esta combinação é interessante para software de sistemas espaciais?

Conformidade

há uma tendência a criação de especificações normatizadas (padronizadas) para implementações de software espacial

funcionalidade

facilidade para reutilização de teste

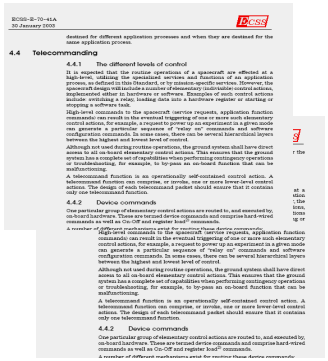
Injeção de falhas

Sistemas a bordo de satélites estão sujeitos a eventos causados por radiação que provocam falhas físicas no hardware (comunicação, memória, processador.)

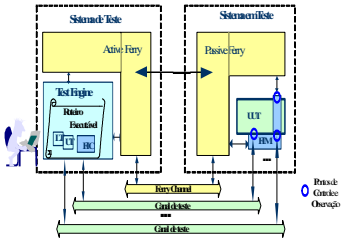
confiabilidade, disponibilidade

Visão geral da CoFI

Documentos em texto
especificações, arquitetura, manual do usuário, etc..



Arquitetura de teste (injeção de falhas)

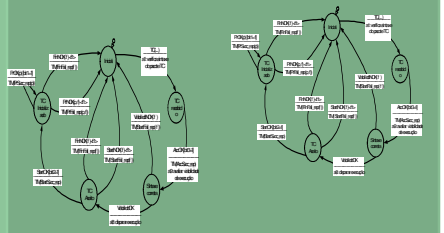


Falhas de Hardware

Serviços

- Initialization
- Diagnostics
- Housekeeping
- Mem.dump

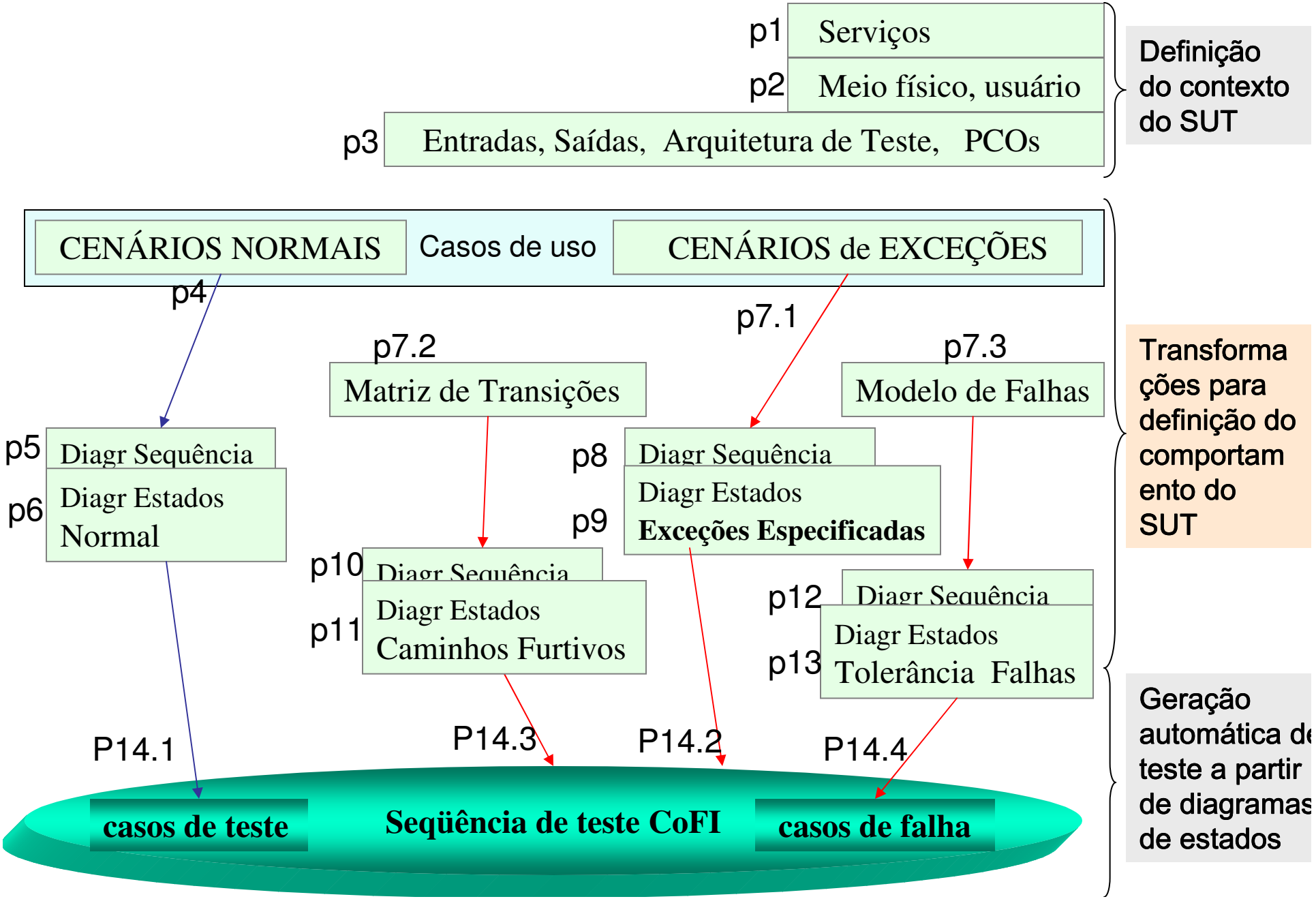
Descrição formal do comportamento em vários modelos



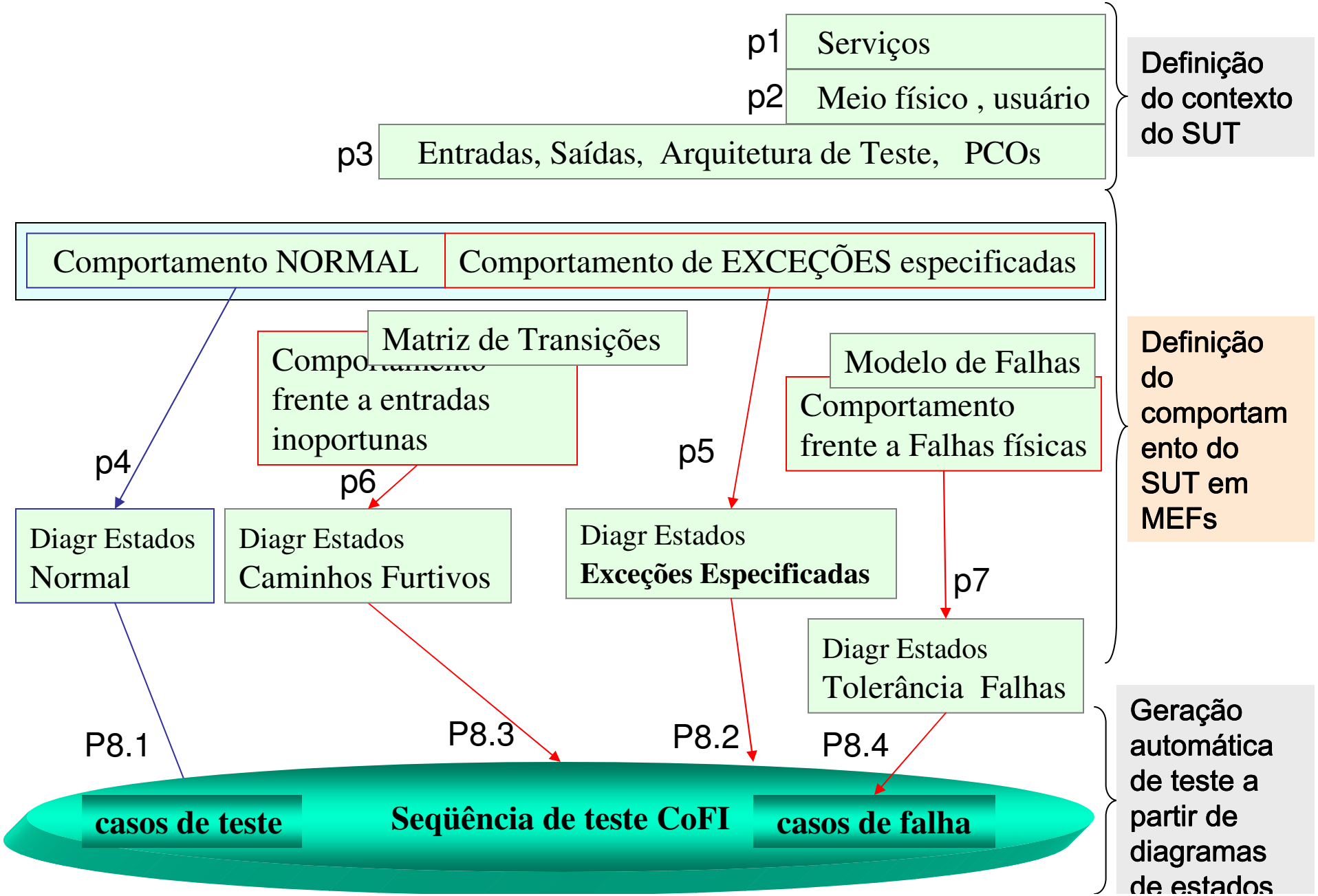
Casos de teste

Casos de falha (injeção falhas)

Passos detalhados da CoFI



Passos simplificados da CoFI



Passos p1 – p3 : definição do contexto do SUT

Identificar:

- **serviços** que um usuário reconhece e pode usar do SUT
- as **falhas físicas** que podem ocorrer no hardware (e que o SUT deveria resistir)
- as facilidades/restrições do **Sistema de Teste**, os **pontos de controle e observação** (PCO), endereços físicos e lógicos, etc..
- os **entradas e as saídas** do SUT

Criação dos modelos parciais

Para cada serviço, definir o comportamento:

- **Normal**
- frente a **Exceções especificadas**
- frente a entradas inoportunas (entradas normais ocorridas em momentos inesperados) (**Caminhos furtivos**)
- frente as falhas de hardware (**Tolerância a falhas**)

Passo p4 : criação de modelos do comportamento **normal**

- identificar a operação normal, rotineira, de uso do serviço
- identificar as entradas e as saídas esperadas para esta operação

o modelo do comportamento normal mostra a seqüência de eventos que o SUT espera normalmente

Se as informações não estiverem contidas nos documentos, o testador deve requisitá-las

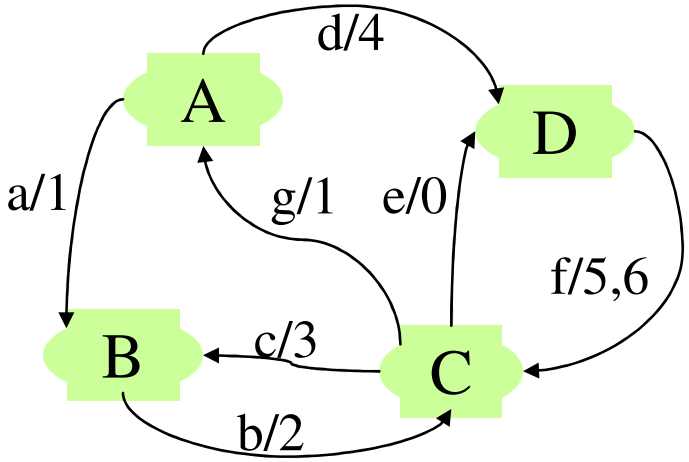
Passo p5: criação de modelos de **exceções especificadas**

- levantar as exceções que foram citadas nos documentos do software
- identificar os entradas e as saídas esperadas neste contexto (definindo assim as ***entradas de exceções***)
- tomar um modelo do comportamento normal do serviço (definido no passo anterior) e modificá-lo:

  (i) incluindo os *eventos de exceções* em novas transições e

  (ii) excluindo caminhos já reconhecidos no passo anterior, mas mantendo o modelo conexo, com estado inicial e final

Passo p6: criação de modelos de **Caminhos Furtivos**



Tomar um modelo normal e escrevê-lo na forma tabular

Identificar as células em branco da tabela

Modificar o modelo normal:

(i) incluindo os *eventos* nos estados onde eles não existiam e

(ii) excluindo caminhos já representados nos passos anteriores

Tabela eventos x estados

	A	B	C	D
a	1/B			
b		2/C		
c			3/B	
d	4/D			
e			0/D	
f				5,6/C
g			1/A	

Passo p7: criação de modelos de **Tolerância a Falhas**

- associar as falhas físicas a ***eventos de falhas***
- Para cada *tipo de falha física* tomar um modelo do comportamento normal e modificá-lo:

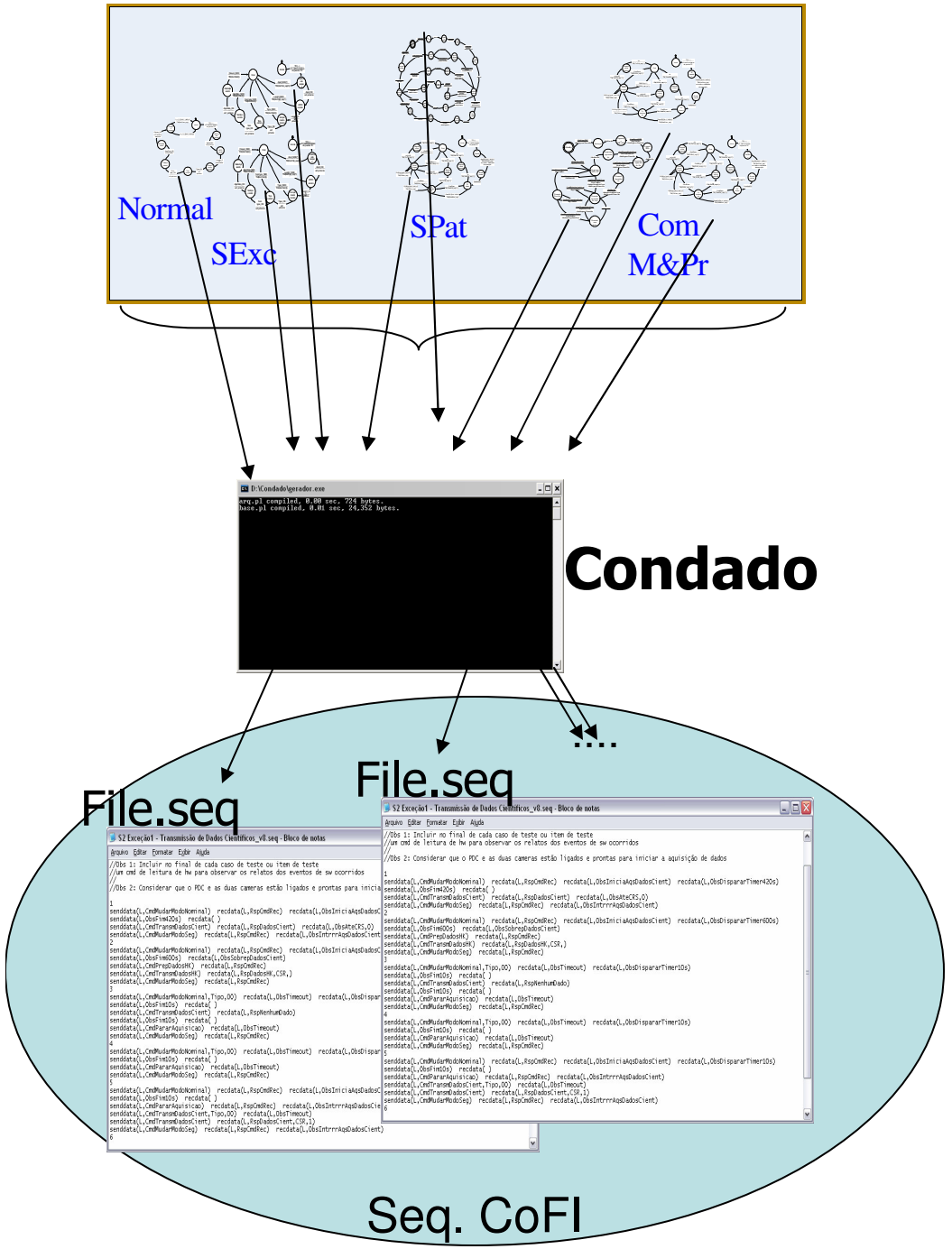
  (i) incluindo os *eventos de falhas* em novas transições e

  (ii) excluindo caminhos já reconhecidos nos passos anteriores, mantendo o modelo conexo, com estado inicial e final

Passo 8: geração automática dos testes

- submeter cada modelo (máquina de estado) à uma ferramenta de geração de casos de teste (Condado [Martins,1999])

  A **Sequência CoFI** consiste da união dos casos de testes gerados para cada máquina



Cada modelo dá origem a um conjunto de casos de teste em um arquivo

Cada arquivo de teste traduz um "objetivo de teste"

Cada "objetivo de teste" contém vários de casos de teste

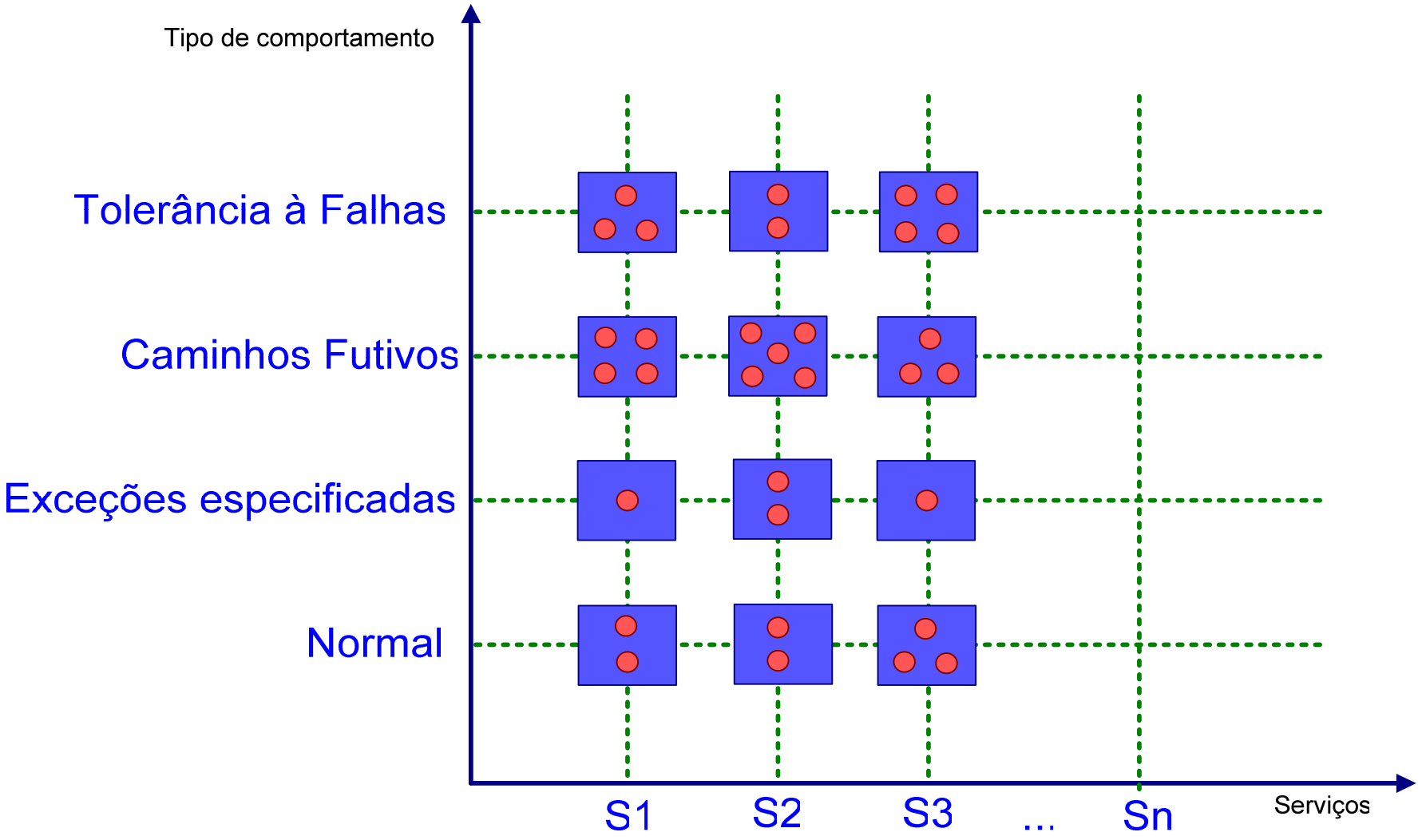
Cada caso de teste contém vários passos de teste

Cada passo de teste contém um evento a ser gerado para o SUT.

Cada evento tem associado a ele, 0 ou várias ações correspondentes que espera-se que sejam produzidas pelo SUT

Modelos parciais do comportamento

Cada modelo (ilustrada na figura com um ponto vermelho) representa um **objetivo de teste**



avaliação da metodologia COFI

Avaliação da eficácia da **Seqüência CoFI**

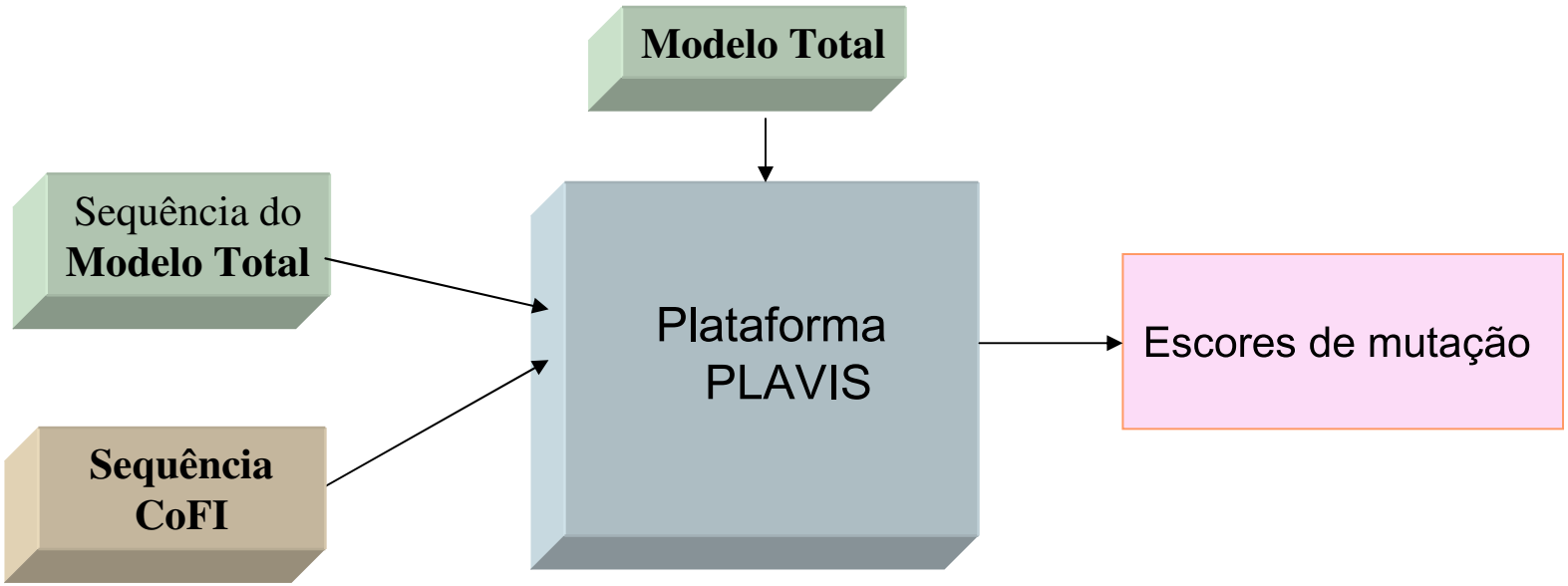
Cálculo do **escore de mutação**
baseado em mutantes de modelo de estados
(plataforma PLAVIS)

Levantamento do **número de erros encontrados**
quando os testes foram executados contra a
implementação

Avaliação pelo escore de mutação

A **Seqüência de teste CoFI** é comparada com a Sequência de teste gerada a partir do **Modelo Total**.

PLAVIS: gera mutantes do Modelo Total, executa as sequências de teste e calcula o escore de mutação.



Exemplo 1- serviço normatizado da área espacial

Avaliação pelo **escore de mutação**

Exemplo 2 - protocolo simplificado para comparação com a abordagem N+[Binder,2000]

Exemplo 3 – SWPDC software de controle de experimento científico

Avaliação pelo **número de erros encontrados** no software

Exemplo 1 – Serviço de verificação de telecomando a bordo de satélites

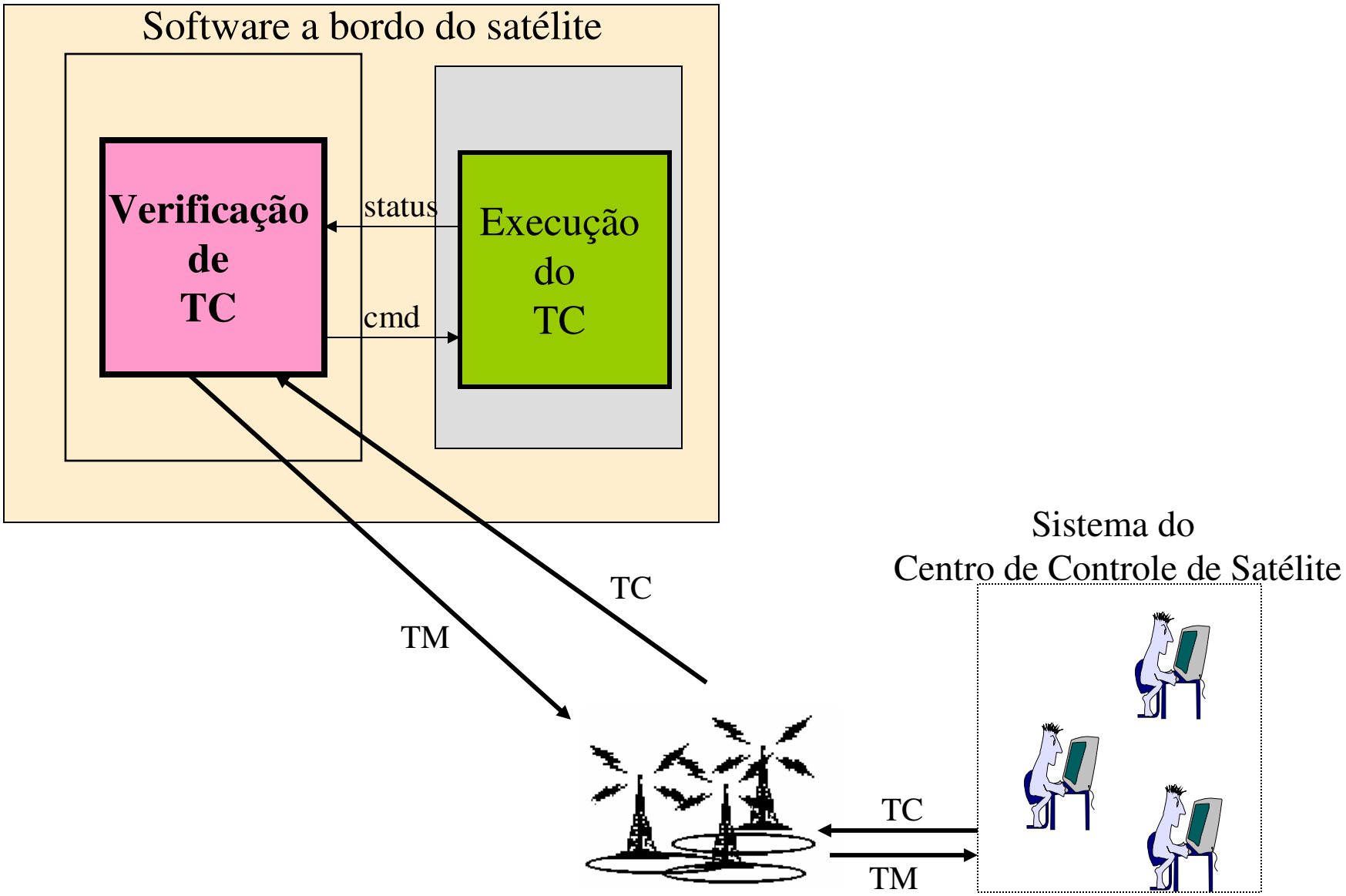
Documento em texto: Norma ESA (European Space Agency) - ECSS-E-7041A - Packet Utilization Service, 2003. Padrão de serviços típicos de computadores a bordo de satélites

Decomposição: detalhada

Avaliação: **escore de mutação**

Experiência: a decomposição do comportamento em 1 e em 2 propósitos de teste

Serviço Verificação de TC



Propósitos de teste, usuários, meio

Propósitos de teste:

1. Checar a correteza do pacote de TC e a viabilidade de sua execução

2. Checar a geração correta de TM sobre o estado da execução do TC para informar o Sistema do Centro de Controle

Usuários:

1. Software no Sistema do Centro de Controle
2. Executor do TC (um processo computacional ou um equipamento)

Meio de Comunicação:

1. Solo-bordo: antenas + protocolo CCSDS
2. Bordo-bordo (não claramente especificada)

Entradas, Saídas, PCOs

Entradas			
	Nome	*PCO*	*Descrição*
	TC	PCO_1	pacote de telecomando.
	StartOK	PCO_3	sucesso no início da execução do TC
	PrOK(p)	PCO_3	sucesso no passo p da execução do TC
	FinOK	PCO_3	sucesso na finalização da execução do TC
	StartNOK	PCO_3	falha no início da execução do TC
	PrNOK(p)	PCO_3	falha no passo p da execução do TC
	FinNOK	PCO_3	falha na finalização da execução do TC
Saídas			
	TM	PCO_2	pac telemetria contendo *reports*.

Diagrama Seqüência Normal

Propósito de teste 1 - correteza do pacote

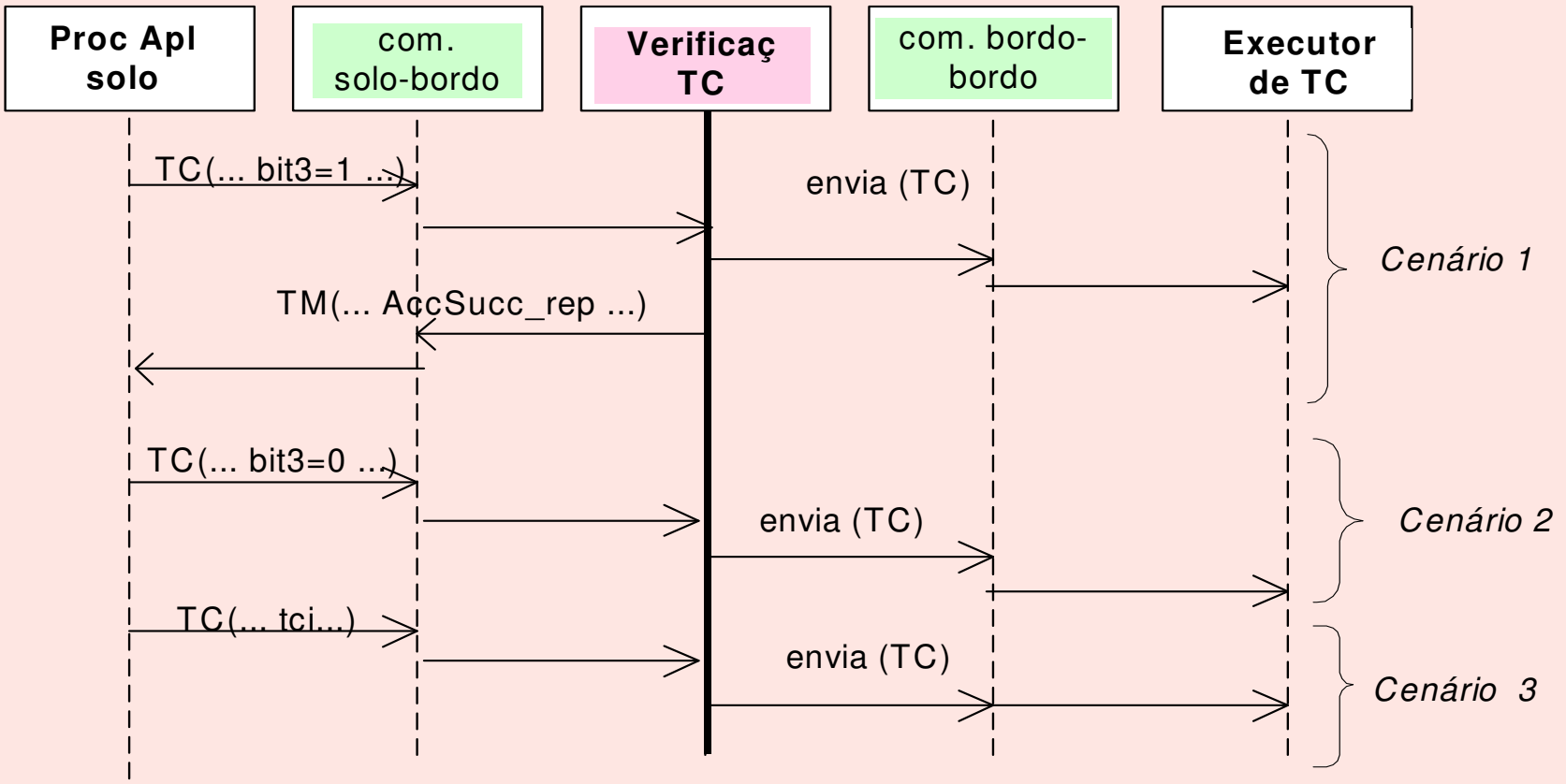


Diagrama Seqüência Tolerância a Falhas

Propósito de teste 2 - execução do TC

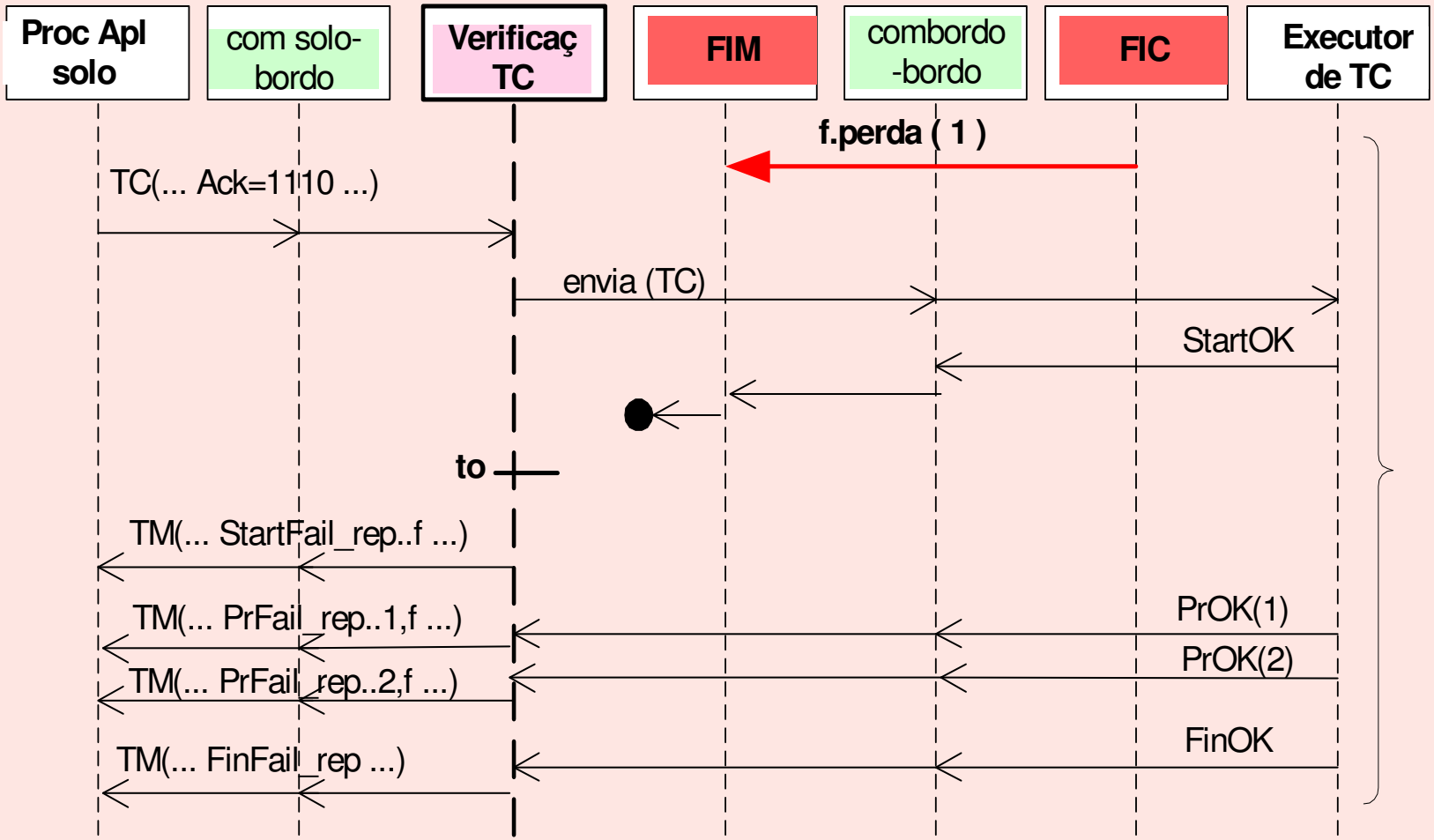
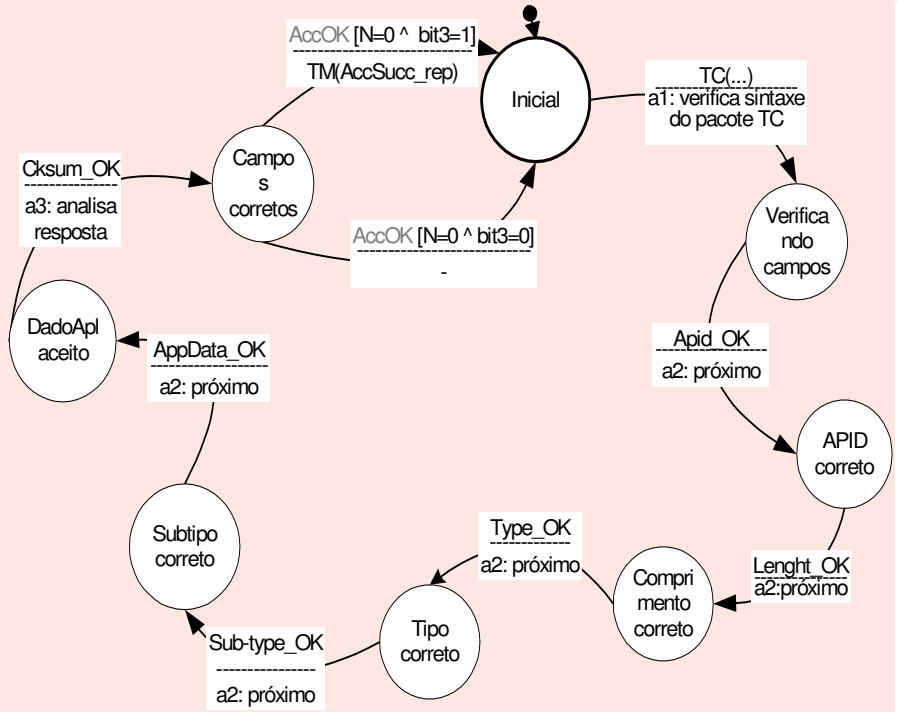


Diagrama de Estado Normal e Casos de Teste

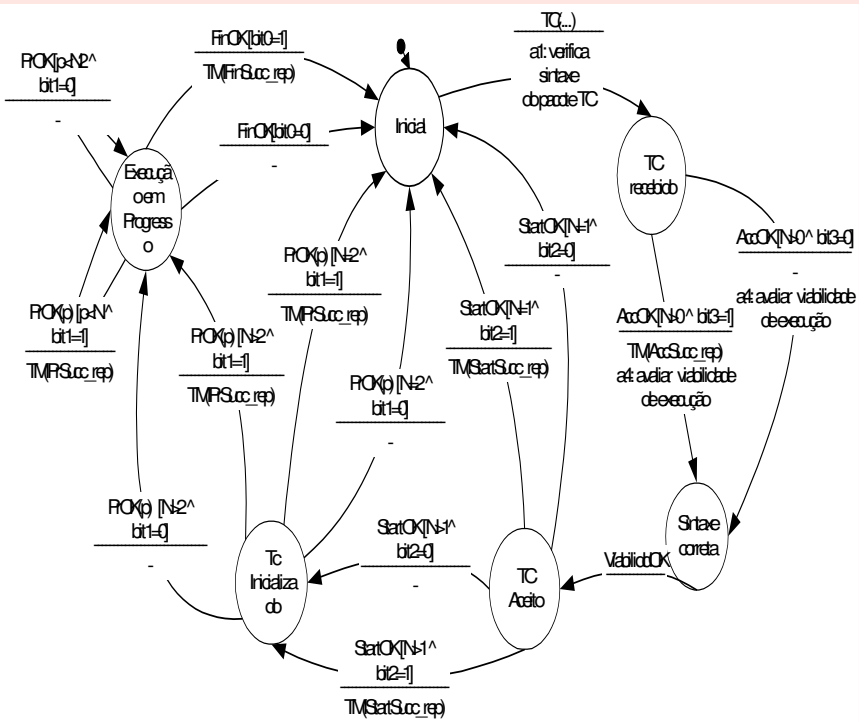
Propósito de teste 1 correteza do pacote



Condado

```
senddata(L,TC)  recdata(L,a1VerificaSintaxe)
senddata(L,ApidOk)  recdata(L,a2Proximo)
senddata(L,LenghtOk)  recdata(L,a2Proximo)
senddata(L,TypeOk)  recdata(L,a2Proximo)
senddata(L,SubTypeOk)  recdata(L,a2Proximo)
senddata(L,AppDataOk)  recdata(L,a2Proximo)
...
```

Propósito de teste 2 - execução do TC



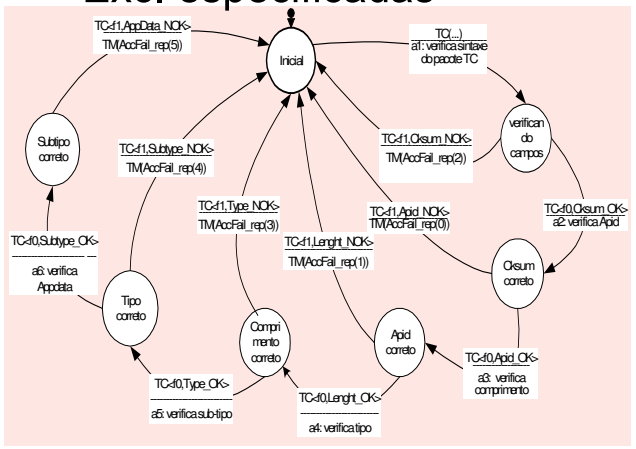
Condado

```
senddata(L,TC)  recdata(L,a1VerificaSintaxe)
senddata(L,AccOKBit3igual1)
recdata(L,TMAccSuccRep)
recdata(L,a9AvaliarViabilidade)
senddata(L,ViabilidadOK)
recdata(L,a8DispararExecucao)
senddata(L,StartOKBit2igual1)
recdata(L,TMStartSuccRep)
...
```

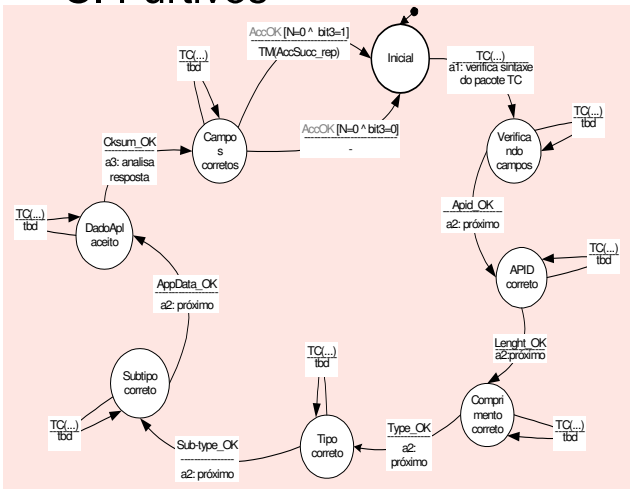
Diagramas de Estados e Casos de falha

Propósito de teste 1

Exc. especificadas

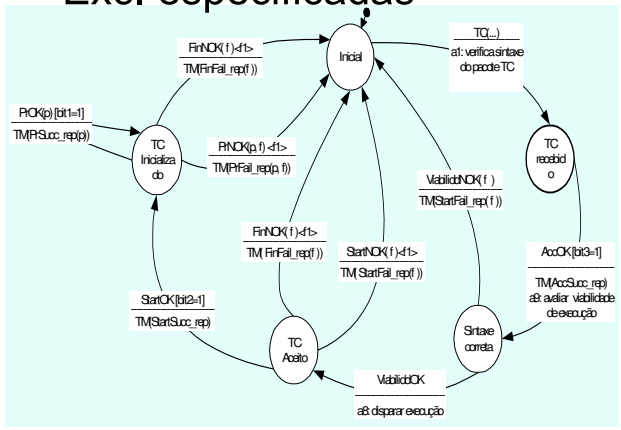


C. Furtivos

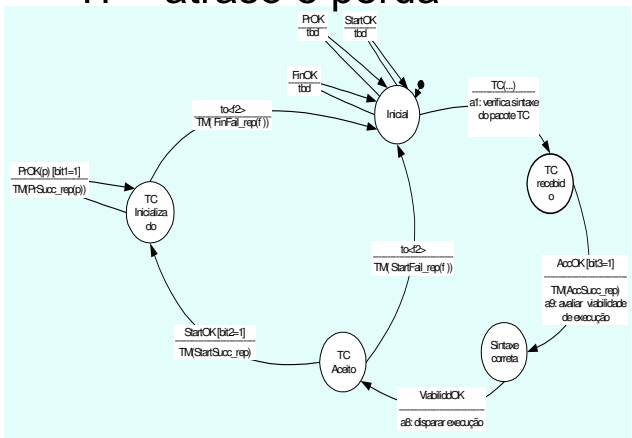


Propósito de teste 2

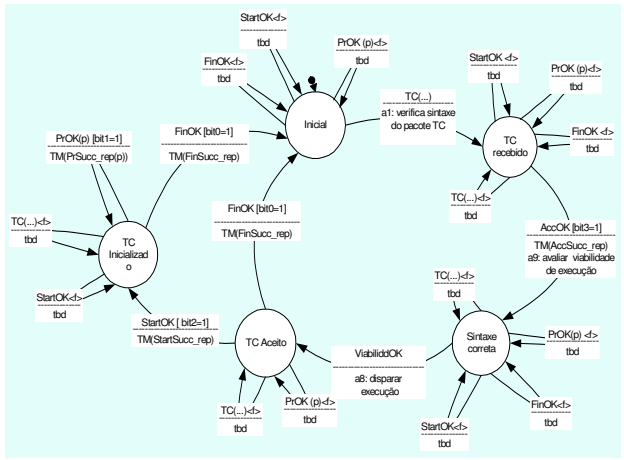
Exc. especificadas



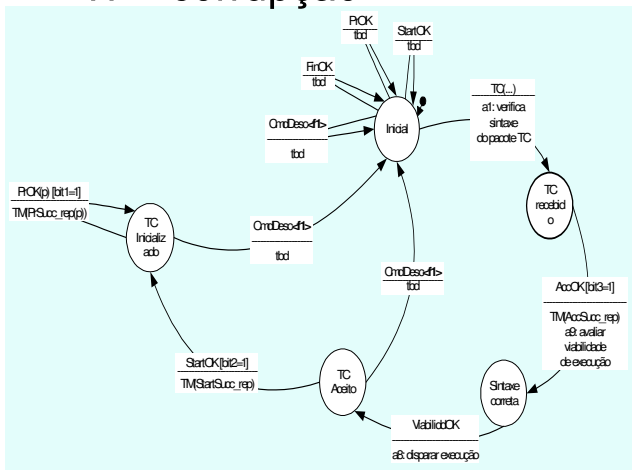
TF - atraso e perda



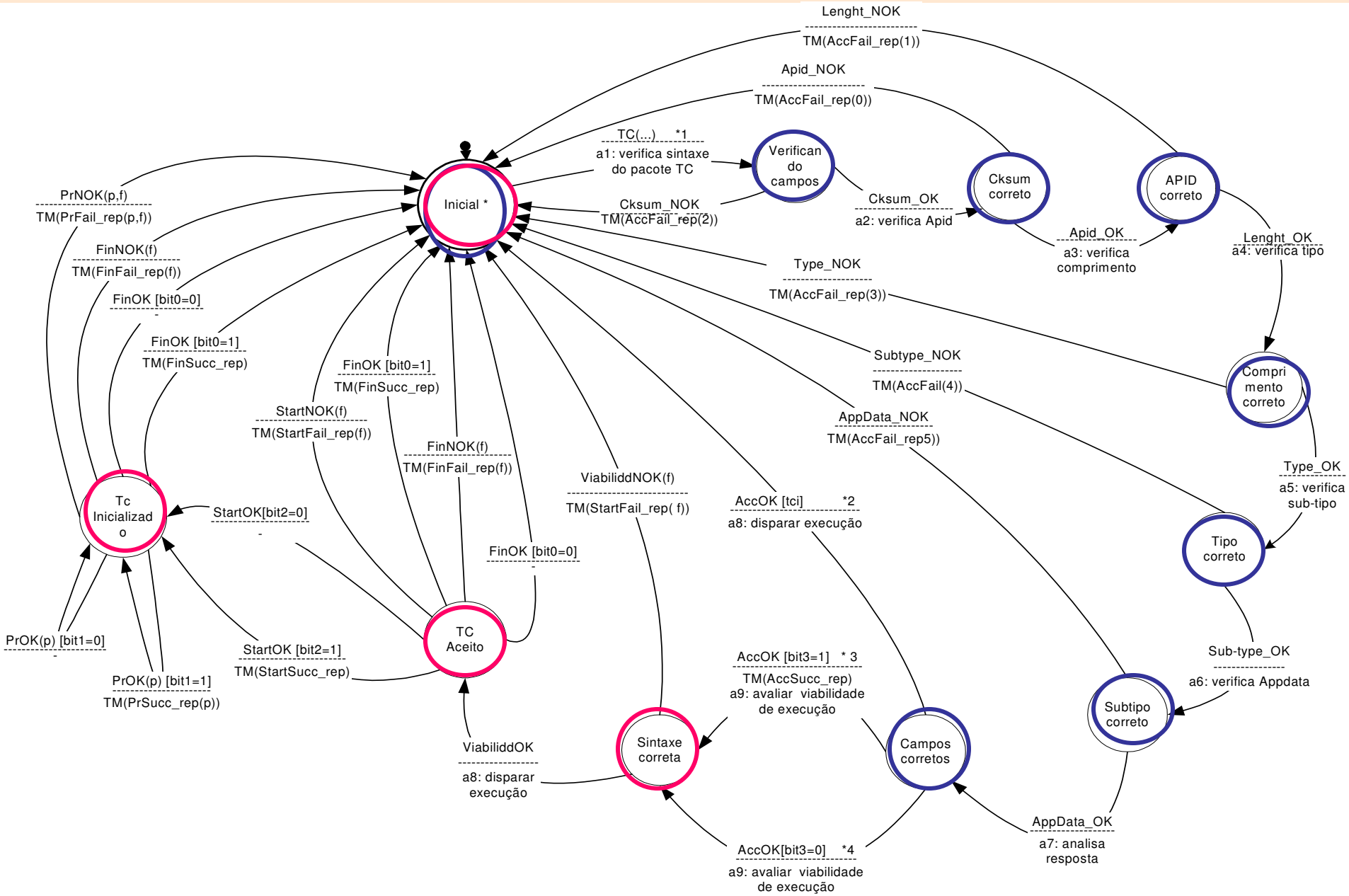
C. Furtivos



TF - corrupção



Modelo Total



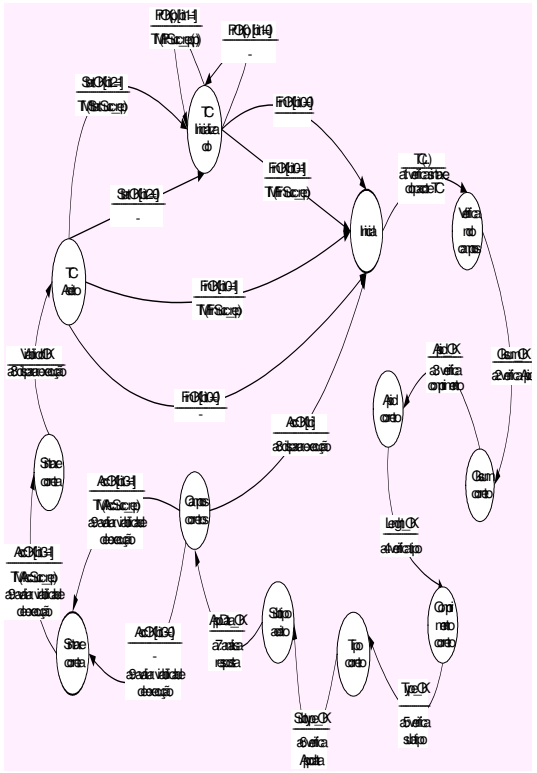
Resultados 1

Para cada propósito de teste foi criado 1 modelo de estado para cada tipo de comportamento.

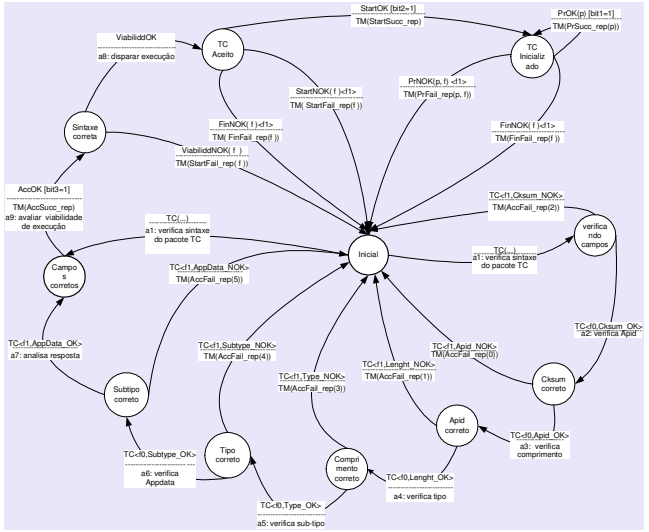
Scofi-propósitos separados	66 casos de teste	0,630 escore
Smodelo total	41 casos de teste	0,978 escore

Diagramas de Estados – propósitos de teste juntos

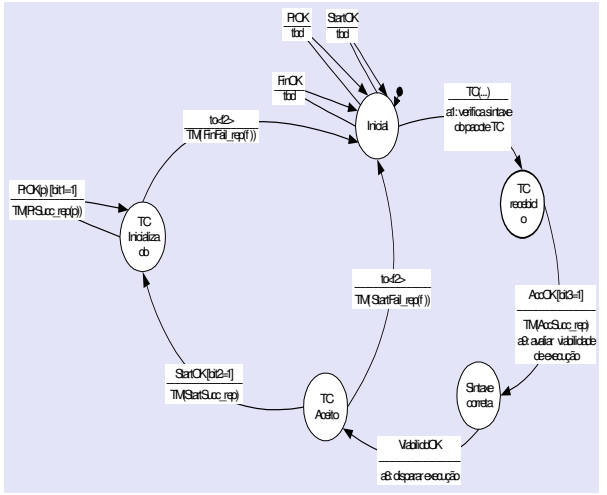
Normal



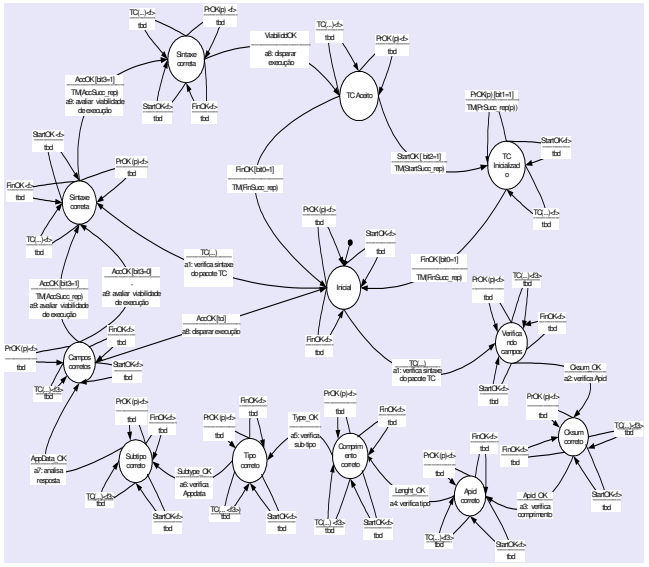
Exc. especificadas



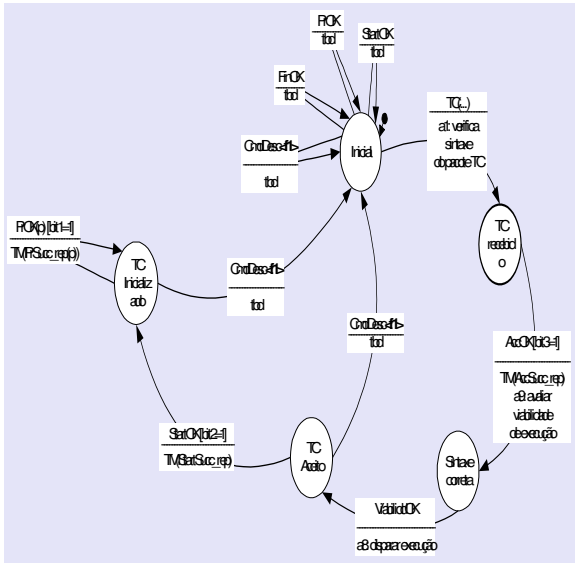
TF - atraso e perda



C. Furtivos



TF - corrupção



Resultados 2

propósitos de teste modelados juntos nos diferentes tipos de comportamento

Scofi-propósitos juntos	107 casos de teste	0,978 escore
Smodelo total	41 casos de teste	0,978 escore

Conclusão 1: quando os modelos parciais contêm estados muito distintos do Modelo Total, os casos de teste gerados a partir dos modelos parciais não cobrem os caminhos do Modelo Total, daí a diferença no escore de mutação.

Conclusão 2: os casos de teste gerados a partir dos modelos parciais são efetivos quando os estados do Modelo Total são, o máximo possível, mantidos nos modelos parciais.

Exemplo 2 – protocolo simplificado

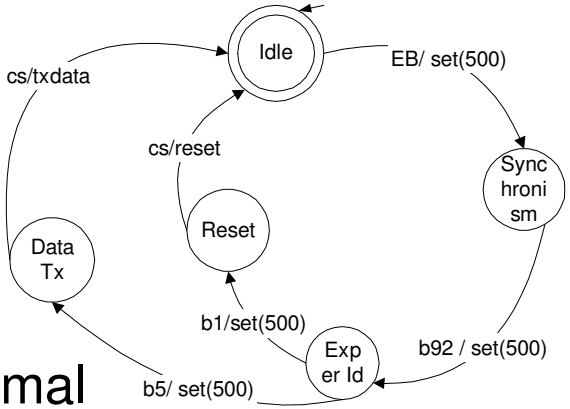
Documento em texto: especificação do protocolo OBDH-EXP (INPE)

Decomposição: simplificada

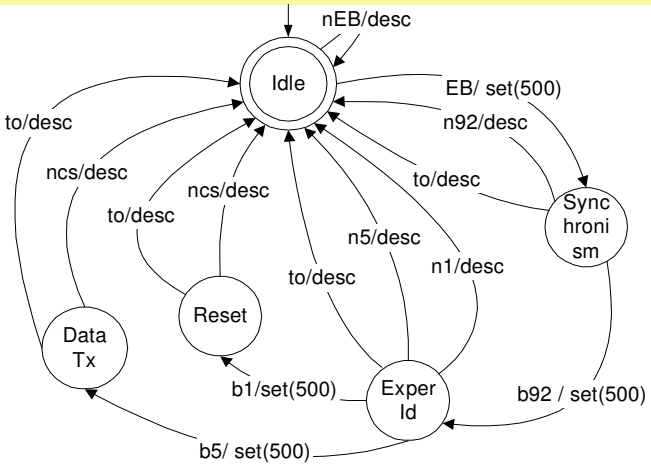
Avaliação: **escore de mutação**

Experiência: comparação com outra abordagem de teste (N+)

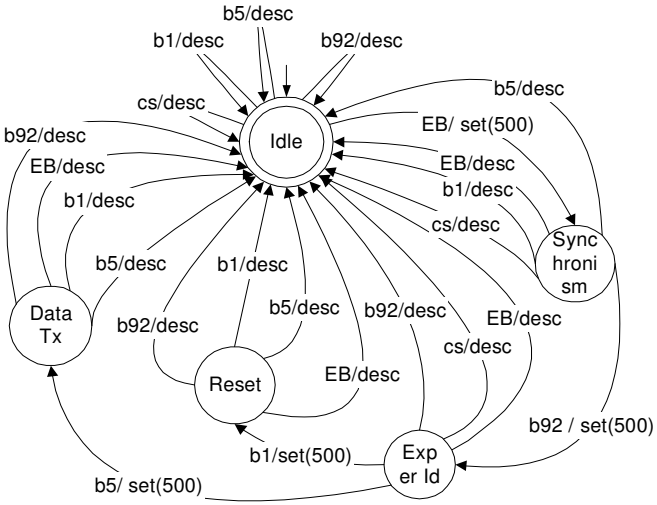
Modelos parciais



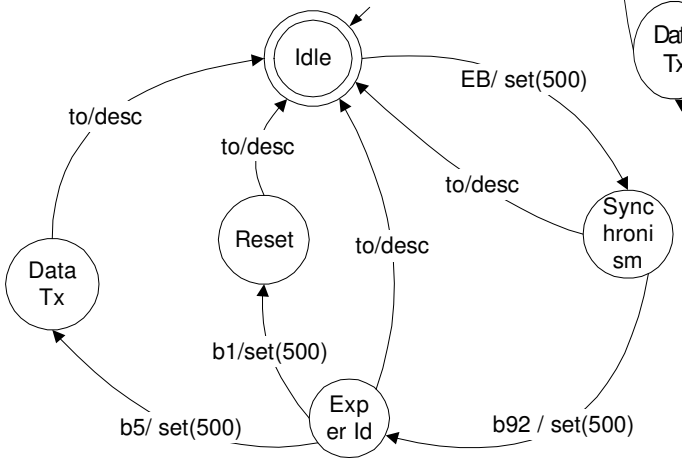
Normal



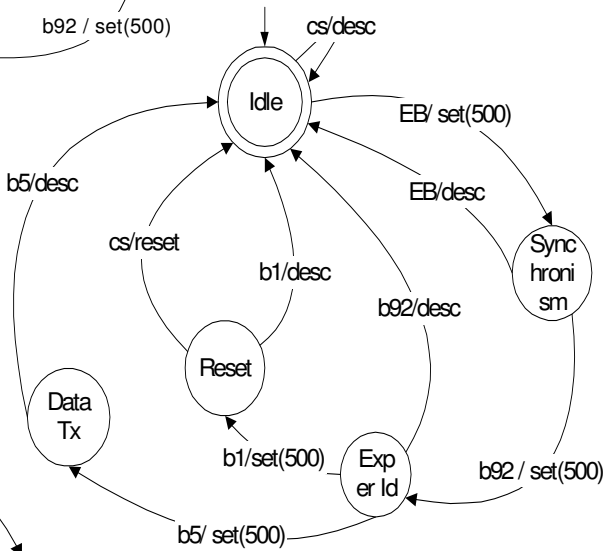
Exc Especificadas



C. Furtivos

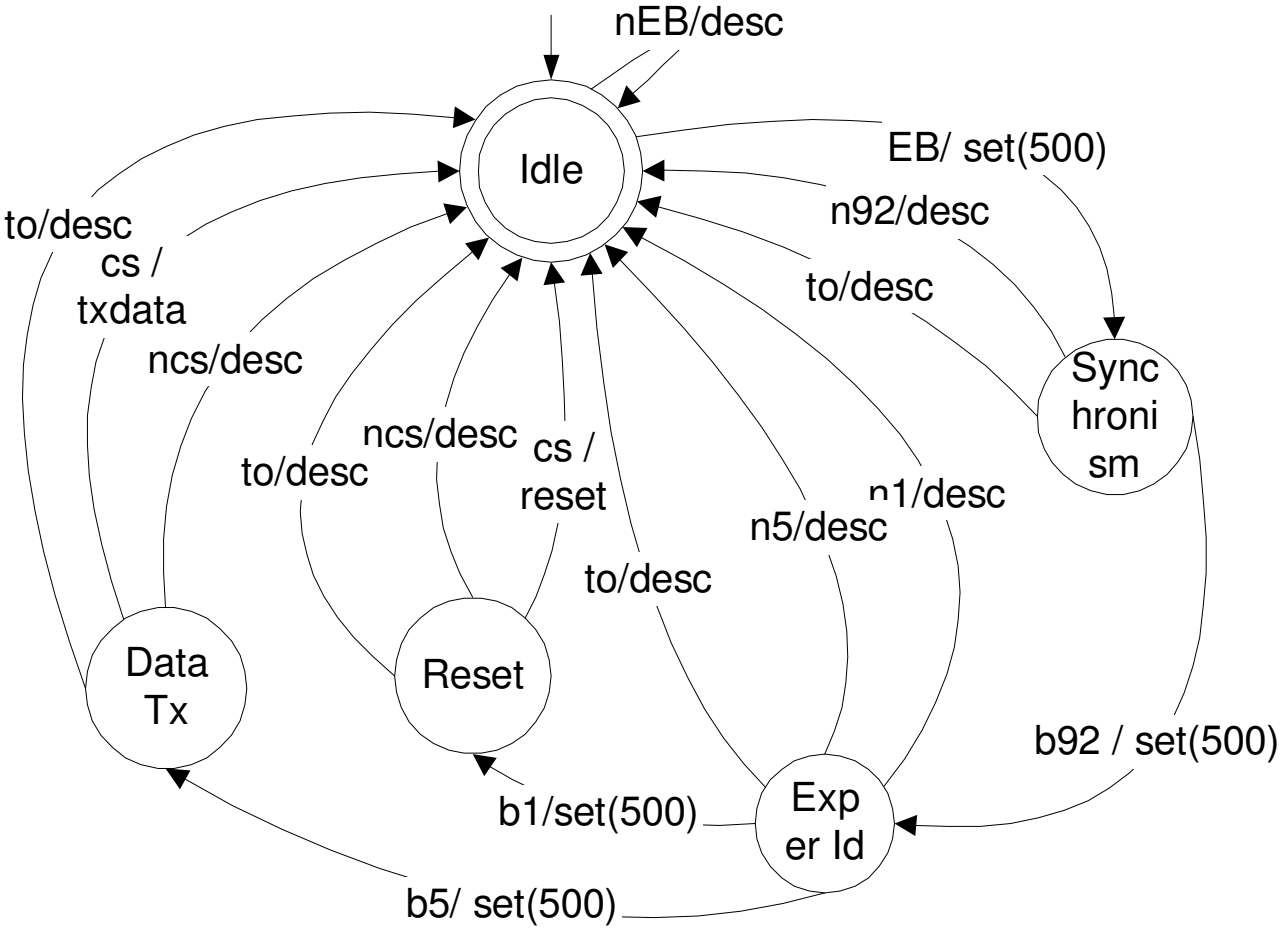


TF - atraso

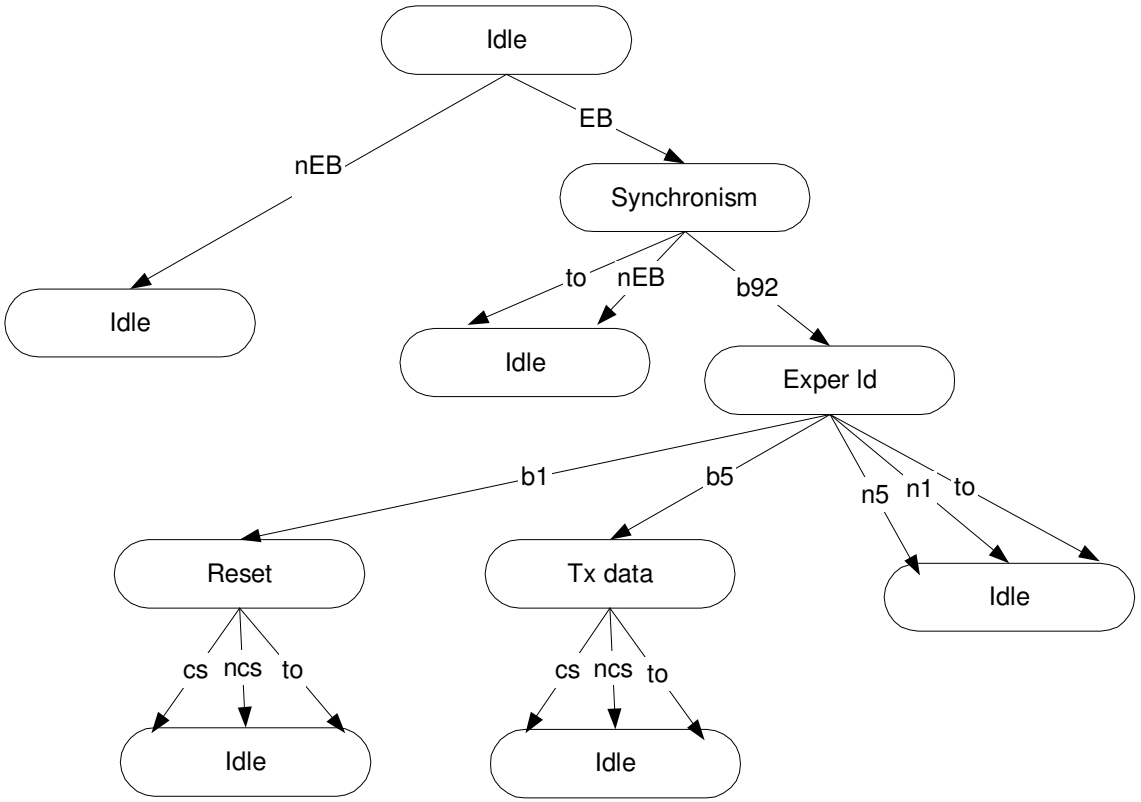


TF - duplicação

Modelo Total



Abordagem N+



Árvore de alcançabilidade

+ Matriz resposta

Resultados da comparação

Seqüência de teste	# c.t + c.f	# c.t. + c.f distintos	# Mortos	# Vivos	Escore
N+	**47**	**46**	154	7	**0,956**
CoFI	**41**	**33**	161	0	**1,000**

Exemplo 3 – SWPDC software de controle de experimento científico

Documentos em texto: especificação, projeto preliminar, projeto detalhado, manual do usuário do software, manual da ferramenta de teste do Projeto QSEE

Decomposição: simplificada

Avaliação: **erros encontrados**

Experiência: CoFI aplicada a **testes de aceitação** em um Processo de Verificação e Validação Independente. O número de erros encontrados na implementação foi computado.

Projeto QSEE

http://www.cea.inpe.br/qsee/

Qualidade de Software Embarcado em aplicações Espaciais

Objetivos:

Transferência de tecnologia do INPE para a indústria nacional de software (DBA)

Definição de um **processo de aceitação** de software para o INPE apoiado na abordagem de Verificação e Validação de Software Independente e no uso da metodologia COFI

Financiadores do projeto QSEE FINEP FINANCIADORA DE ESTUDOS E PROJETOS FUNDEP

Sistema em teste = SWPDC

software embarcado em um **C**omputador de manipulação **D**ados de **P**ayload
que controla experimentos científicos a bordo do
satélite MIRAX em desenvolvimento no INPE

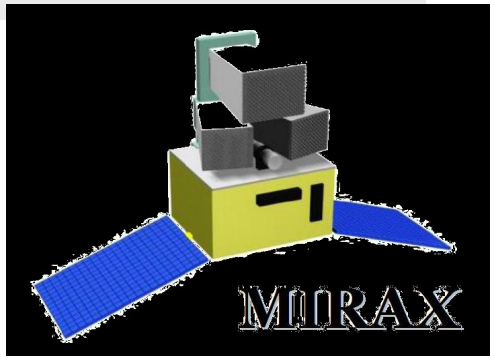
linguagem C; Labview

Principais funções:

- Adquirir dados científicos
- Preparar e transmitir dados de: housekeeping, teste e diagnóstico
- Reconhecer falhas de: comunicação, memória e processador
- Implementar mecanismo de manipulação e tratamento de falhas
- Implementar os protocolos de comunicação com: OBDH, os processadores de eventos (EPPs)

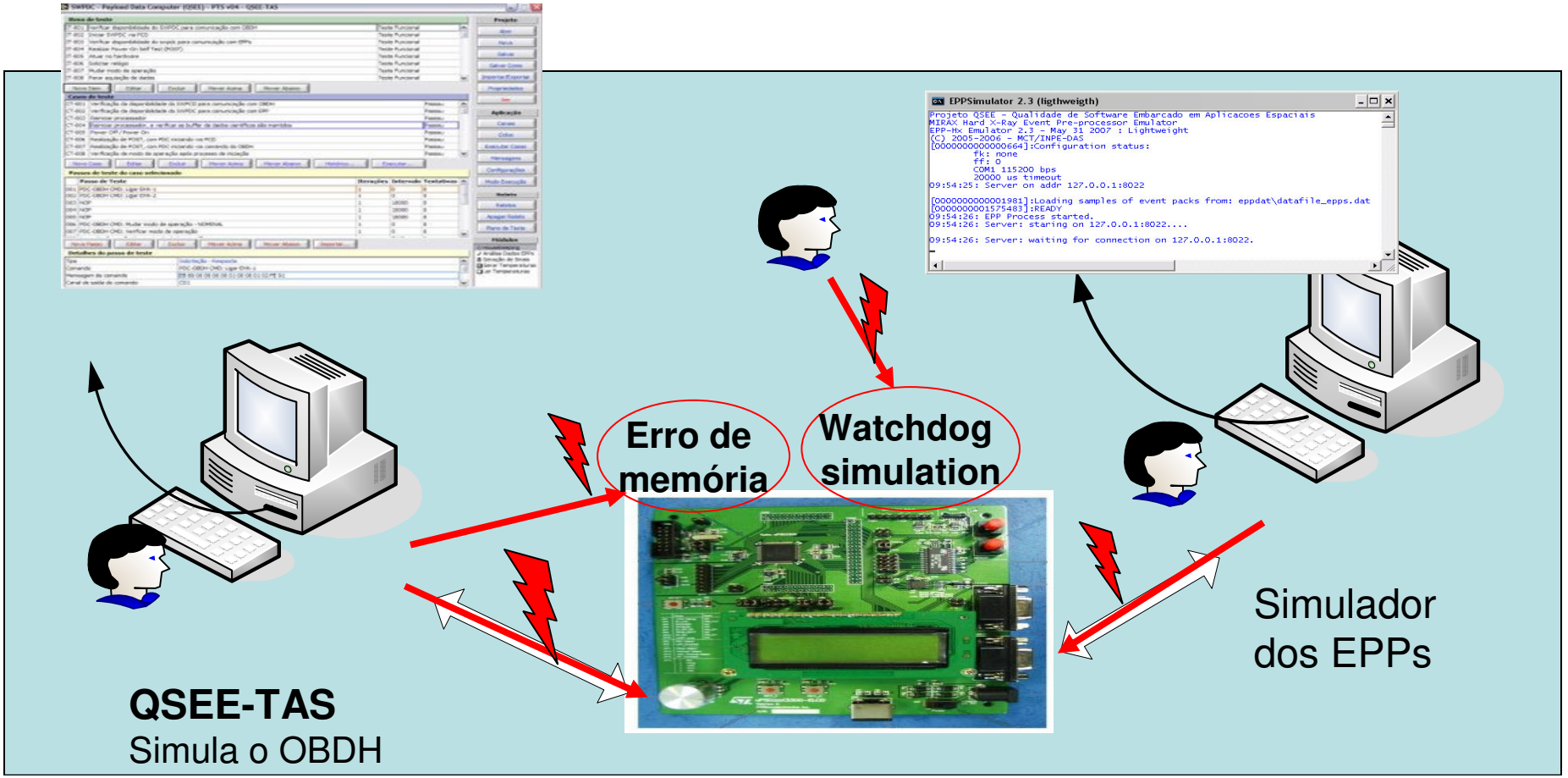
Processor Family : Intel 8051 microcontroller (**uPSD-3334-40D - 8034**
MCU Clock : 40 MHz, Internal Memory : 256 Bytes
External Memory: SRAM: 8 KB; FLASH: 256 KB
Special regs.: 256 Bytes

Code Lines **1970** Comment Lines 6522

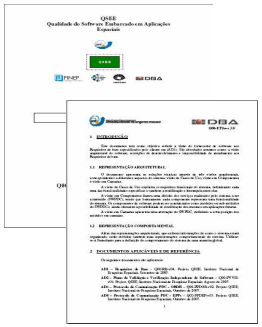


Sistema de Teste

- Ferramentas de teste desenvolvidas pela equipe do INPE
  - QSEE-TAS,
  - Simulador EPPs



Aplicação da COFI



Serviços

- S1
- S2
- S3
- ...
- S11

Normal (Norm)

Exceções Especificadas (SExc)

Caminhos Furtivos (SPat)

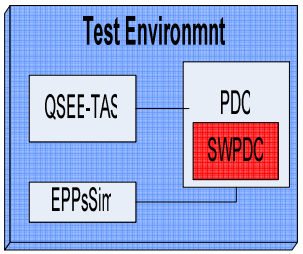
Tolerância Falhas
- comunicação (Com)
- memória → (M&Pr)
- processador → (M&Pr)

Entradas/Saídas

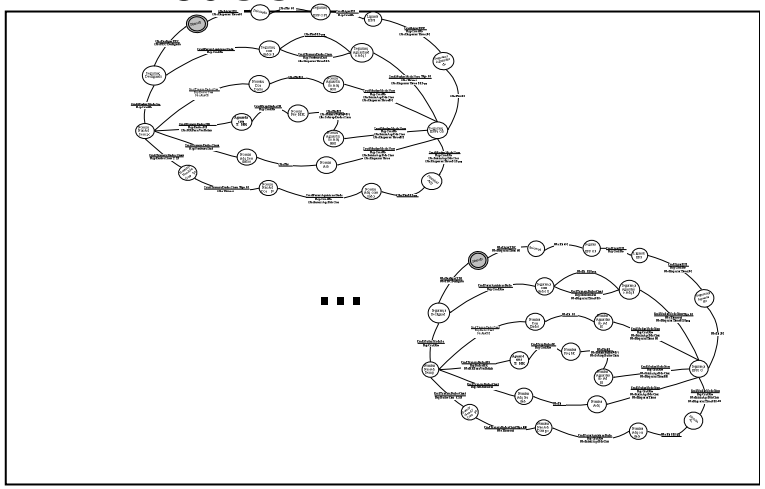
sintaxe das entradas esperadas

comandos para os injetores de falha

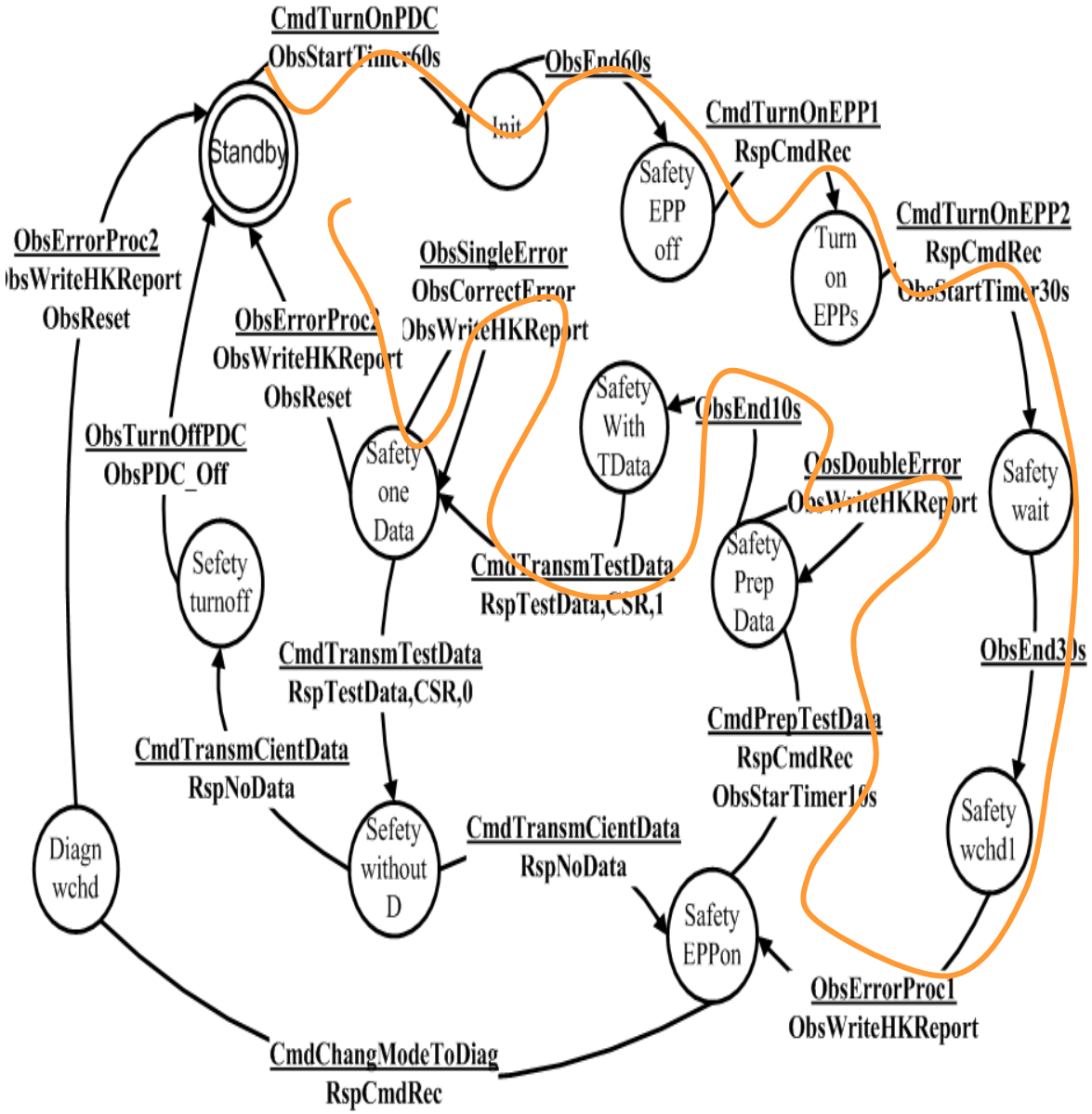
Parâmetros: tipos de falha, timers, ...



Models



Exemplo – comportamento TF - M&Pro



Entrada	Saída Esperada
CmdTurnOnPDC	ObsStartTimer60s
ObsEnd60s	
CmdTurnOnEPP1	RspCmdRec
CmdTurnOnEPP2	RspCmdRec ObsStartTimer30s
ObsEnd30s	
ObsErrorProc1	ObsWriteHKReport
CmdPrepTestData	RspCmdRec ObsStarTimer10s
ObsDoubleError	ObsWriteHKReport
ObsEnd10s	
CmdTransmTestData	RspTestData,CSR,1
ObsSingleError	ObsCorrectError ObsWriteHKReport
ObsErrorProc2	ObsWriteHKReport ObsReset

Códigos interpretados pela QSEE-TAS

Total de serviços e modelos criados

	Norm	SExc	SPat	Comm	M&Pro	Total
Initialization	2	1	1	1	1	6
Scientific data	2	2	1	1	1	7
Housekeeping	3	3	3	1	1	11
Test data	2	4	4	1	1	12
Diagnostics	2	4	4	2	2	14
Memory dump	5	3	5	2	1	16
Change operat mode	1	0	0	0	1	2
Load & exec program	1	5	4	3	2	15
OBDH msg syntax	1	0	0	1	0	2
EPPs msg syntax	1	0	0	1	0	2
Special commands	4	0	0	2	4	10
System load	3	0	0	0	0	3
Service combination	1	1				2
Total	24	24	22	13	14	99

Numero de casos de teste que detectaram erros

Serviços	# casos de teste	# erros	# Non-Conformance Reports
Initialization	46	2	2
Scientific data	98	40	2
Housekeeping	109	0	0
Test data	123	6	2
Diagnostics	69	16	4
Memory dump	119	8	6
Change operation mode	42	0	0
Load & execute program	29	10	6
OBDH message syntax	64	10	10
EPPs message syntax	18	0	0
Special commands	52	12	4
System load	3	0	0
Total	771	104	36

Principais erros detectados pelos Testes de Aceitação

Pedido de transmissão de dados durante a fase de inicialização do sistema era aceito e não deveria.	Comando de dump de memória de uma página inexistente era aceito e não deveria.	Execução de programa não carregado na memória era permitida e não deveria.

Comentários sobre os resultados da execução da Sequência COFI

Revelaram erros em

- Código (45%)
- Documentos (33%)
- Codigo + Documentos (22%)

apesar das revisões realizadas ao longo do desenvolvimento do SWPDC **e dos testes realizados pelo fornecedor**

Dos erros detectados, **76%** foram revelados pelos casos de falhas $\Rightarrow$ evidencia que os casos de falha são de extrema importância para aceitação do sistema.

Lições aprendidas

O esforço de criar os 99 modelos foi compensado pela superior organização dos testes alcançada com a aplicação da metodologia CoFI em comparação com o projeto dos testes criados pelas equipes de desenvolvimento, sem a aplicação de uma metodologia.

Os modelos focam a atenção dos testadores para falhas e exceções que podem ocorrer durante a operação do software levando ao projeto de situações que os desenvolvedores normalmente não pensam.

Os casos de teste gerados pela ferramenta Condado são auto-contidos, i.é, cada caso pode ser executado independentemente, desta forma o veredicto não depende da ordem de execução dos casos e facilita a re-execução.

A metodologia CoFI

Reduz a distância entre a prática (geração de teste a partir de uma especificação textual) e o uso de métodos formais (especificação em autômatos)

Possibilita reuso em teste, porque a geração parte da especificação

Orienta a definição de experimentos determinísticos para Injeção de Falhas

Referências

Martins, E.; Sabião, S.B.; Ambrosio, A.M. - "ConData: a Tool for Automating Specification-based Test Case Generation for Communication Systems" – Software Quality Journal, Vol. 8, No.4, pages 303-319, 1999. Edited by Anna Liu and Paddy Nixon - Kluwer Academic Publishers.

Binder, R. **Testing object-oriented systems** - models, patterns and tools**.** Boston: Addison-Wesley, 2000. 1191p.

Lai, R. A survey of communication protocol testing*.* **The Journal of Systems and Software**, n. 62, p. 21-46, 2002

.

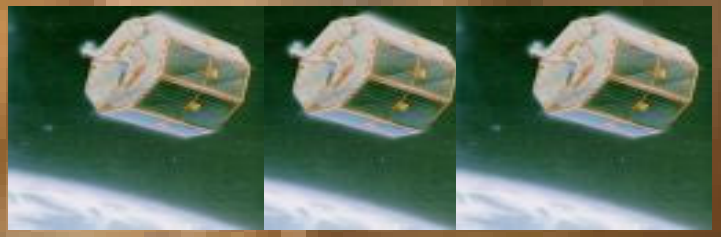
Ambrosio, A. M. CoFI – uma abordagem combinando teste de conformidade e injeção de falhas para validação de software em aplicações espaciais. **Tese de doutorado**, INPE, São José dos Campos, 2005

Ambrosio, A.M.; Mattiello-Francisco, F.; Santiago, V. A.; Silva, W.P.; Martins, E. Designing Fault Injection Experiments using State-based Model to test a Space Software. Third Latin-American Symposium on Dependable Computing (LADC), Morelia, México, Sep. 26-28, 2007. **Lecture Notes in Computer Science (LNCS) series**. Springer Editors: Bondavali, A.; Brasileiro, F.; Rajsbaum, S. Springer, Berlin. 2007; pages 170-178.

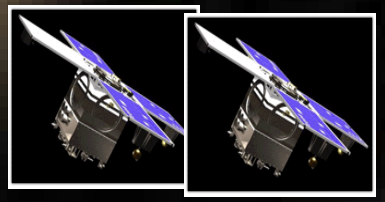
Família de Satélites Brasileiros

CBERS – China-Brazil Earth Resource Satélites

SCD – Satélites de Coleta de Dados

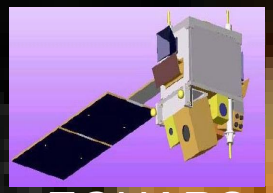


SACI



Scientific Satellites: Satec, ItaSat,

MIRAX– X-ray Imagiator



EQUARS